UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/870,620	05/31/2001	Scott J. Broussard	AUS920010262US1	1774

35617 7590 04/06/2007

DAFFER MCDANIEL LLP
P.O. BOX 684908
AUSTIN, TX 78768

EXAMINER	
BONSHOCK, DENNIS G	
ART UNIT	PAPER NUMBER
2173	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
2 MONTHS	04/06/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/870,620
Filing Date: May 31, 2001
Appellant(s): BROUSSARD, SCOTT J.

Mollie E. Lettang (Reg. No. 48,405)
For Appellant

EXAMINER'S ANSWER

This is in response to the supplemental appeal brief filed 4-20-2006 and the Summary Of Claimed Subject Matter, submitted on 11-22-2006.

(1) Real Party in Interest

A statement identifying the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The following are the related appeals, interferences, and judicial proceedings known to the examiner which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal:

- 09/870,613: Notice of Appeal filed 2/7/05; Appeal Brief filed April 7, 2005.
- 09/870,614: Notice of Appeal filed 10/26/04; Appeal Brief filed December 20, 2004.
- 09/870,615: Notice of Appeal filed 9/14/04; Appeal Brief filed November 9, 2004.
- 09/870,621 : Notice of Appeal filed 9/24/04; Appeal Brief filed November 23, 2004.
- 09/870,622: Notice of Appeal filed 8/24/04; Appeal Brief filed October 25, 2004.
- 09/870,624: Notice of Appeal filed 5/23/05; Appeal Brief filed July 25, 2005.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

No amendment after final has been filed.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Nelson, Matthew T., Java Foundation Classes, 1998, McGraw-Hill, xxv-xxvii, 20-22, 43, 73-79, 472-481, 694-707

6,005,588	***Guha***	***12-1999***

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 3-10, and 12-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nelson, *Java Foundation Classes*.

6. With regard to claim 1, Nelson teaches, a software component that creates a graphical representation of a object, where a graphical representation is rendered, comprising text and

other displayable content (see page 694 and 697), an application program creating a graphical representation under an operating system (see page 20 paragraph 1 and page 39), a second software component adapted for drawing text (see page 472), and this second software invoked to draw text using only visual attributes, i.e. static text (see page 472). Nelson further teaches on page 78, UI class having separate groups of code to get the look-and-feel, and to draw the text, it teaches a UI class that doesn't know what the text control contains or what the contents should look like, but uses getDocument() or getStyledDocument() methods. It would have been obvious to one of ordinary skill in the art, having the teachings of Nelson that the JtextField and Jlabel components could be combined, to allow for one components to handle the look-and-feel and one component to handle the displaying of the text.

7. With regard to claims 3 and 12, which teach the first software component used in a graphical representation of an object, Nelson teaches, on page 694, a graphical representation of the text, and further teaches on page 472, and on page 78, a second software component used to draw the text defined by another software component where the second software component is not aware of what text it contains or what it looks like.

8. With regard to claims 4 and 13, which teach the first software component being adapted to support undo and redo editing of the text content in the graphical representation of the object, Nelson further teaches, on page 696, text being inserted or removed from a JtextField.

9. With regard to claims 5 and 14, which teach the object being part of a graphical interface associated with the application program, Nelson further teaches, on page xxv, Java Foundation Classes, which Swing (JtextField and Jlabel) is, help programmers make their GUIs slightly faster than before, and help functionality and appearance.

10. With regard to claim 6, which teaches the application program written in Java programming language, Nelson further teaches, on page xxv, Java Foundation Classes, which Swing (JtextField and Jlabel) is, help programmers make their GUIs slightly faster than before, and help functionality and appearance.

11. With regard to claim 7, which teaches the first and second components comprising a Java virtual machine of Swing application program interface, Nelson further teaches, on page 81, Java Foundation Classes, which Swing (JtextField and Jlabel) is, Accessibility *Application Programming Interface* (API) is to make it easy for programmers to make their applications work with access technologies that can sit on top of the *Java Virtual Machine.*

12. With regard to claim 8, Nelson teaches, an application program running under an operating system (see page 20 paragraph 1 and page 39) comprising: a software component the creates a graphical representation of a object, where a graphical representation is rendered, comprising text and other displayable content (see page 694 and 697), a second software component adapted for drawing text (see page 472), and this second software invoked to draw text using only visual attributes, i.e. static text (see page 472). Nelson further teaches, on page 78, UI class having separate groups of code to get the look-and-feel, and to draw the text it teaches a UI class that doesn't know what the text control contains or what the contents should look like, but uses getDocument() or getStyledDocument() methods. It would have been obvious to one of ordinary skill in the art, having the teachings of Nelson that the JtextField and Jlabel components could be combined, to allow for one component to handle the look-and-feel and one component to handle the displaying of the text.

13. With regard to claim 9, which teaches executing comprising creating a label upon the display absent any text, Nelson further teaches, on page 473, creating a label that is absent text.

14. With regard to claim 10, which teaches executing comprising creating a border upon the display absent any text within the border, Nelson further teaches, on page 697, paragraph 2, creating an EmptyBorder on the display.

15. With regard to claim 15, which teaches using a second software component (as shown supra) to establish an appearance and behavior of he object being independent on the operating system, Nelson further teaches, on page 43, "metal" which gives Swings universal Look-and-feel.

16. With regard to claim 16, Nelson teaches, teaches the appearance an behavior of he object being independent on the operating system (see page 43), a windows based operating system (page xxvii), an application program running under an operating system (see page 20 paragraph 1 and page 39) comprising: a software component the creates a graphical representation of a object, where a graphical representation is rendered, comprising text and other displayable content (see page 694 and 697), a second software component adapted for drawing text (see page 472), and this second software invoked to draw text using only visual attributes, i.e. static text (see page 472). Nelson further teaches on page 78, UI class having separate groups of code for getting the look-and-feel, and for drawing the text, it teaches a UI class that doesn't know what the text control contains or what the contents should look like, but uses getDocument() or getStyledDocument() methods. It would have been obvious to one of ordinary skill in the art, having the teachings of Nelson that the JtextField and Jlabel components could be combined, to

allow for one component to handle the look-and-feel and for one component to handle the displaying of the text.

17. With regard to claim 17, which teaches a peer component for redirecting a memory call to invoke text drawing methods of the second software component rather than text drawing methods of the first software components, Nelson teaches, on pages 694, 697, 472, and on page 72, a system for drawing text where one component can define attributes of an item and the actual displaying of the item can be implemented by another item. It is, however a design choice, an item could just as easily be given attributes and drawn by the same software component.

18. Claims 2 and 11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nelson, *Java Foundation Classes*, and Guha, patent #6,005,588.

19. With regard to claims 2 and 11, which teach the operating system assigning text drawing placed in a buffer and the buffered text edited prior to being drawn, Nelson teaches the system for defining and displaying text using separate components (as rejected supra), but doesn't teach assigning text drawing placed in a buffer and the buffered text edited prior to being drawn. Guha teaches a system of rapidly displaying text as did Nelson but further teaches, in column 2, lines 14-45, placing characters in a bitmap in memory, where there pixels are combined into lines to reduce the amount of memory space required to store. It would have been obvious to one of ordinary skill in the art, having the teachings of Nelson and Guha before him at the time the invention was made to modify the system for defining and displaying text using separate components of Nelson to include the buffer and editing before being displayed, as did Guha.

One would have been motivated to make such a combination because this greatly improves the speed and memory usage of displaying text.

(10) Response to Argument

Claims 1, 3-10, and 12-16:

With respect to the group of claims including Claims 1, 3-10, and 12-16, the Appellant's arguments are focused on the limitations regarding a first software component for creating a graphical representation of an object, and define visual attributes, and a second software component that draws the text. More specifically, as stated from representative Claim 1, the limitation argued is:

> *a first software component adapted to create a graphical representation of an object embodied as code within the software component, wherein the code comprises text and other displayable content;*
>
> *an application program running under an operating system; and*
>
> *a second software component adapted for drawing the text, wherein the first software component is invoked during runtime by the application program to define visual attributes of the text, but not to draw the text, and wherein the second software component is invoked to draw the text using the visual attributes.*

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The Examiner asserts the limitation to be a computer

readable medium in which a first software component defines visual attributes of text, but does not draw the text; a second software component then draws the defined text.

As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

> "*Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure.* In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997)."

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the references, are within the scope of these limitations.

- *Nelson teaches a system in which a UI class (second software component) draws text components, but doesn't know what the text control contains or what the content s should look like, so it gets the Document using the Text Component's getDocument() or getStyledDocument() method (first software component) (see page 78, under the "Interacting with Documents" heading). Nelson further teaches Documents containing Elements that comprise a mapping of Document characters to attributes, these attributes defining how a document text should look (see page 73 all). The "Interacting with the UI" heading further shows that "the UI class retrieves each Element from the Document and creates a View that knows how to draw each one."*

Before answering individual argument the examiner would like to point out several points that the applicant has admitted to in the appeal brief and prior submission.

- *the Amendment filed on 5-17-04, on page 11, in the first paragraph, the applicant points out that "The Applicant agrees that separate groups of Swing-based code may be used for setting the look-and-feel of a displayed object and for drawing text within or upon the displayed object. For example, the "setLookAndFeel" method is commonly used to define the look-and-feel of a text field (and even the associated text) drawn by the JtextField Component." To this the examiner agrees that separate groups of code can be used for setting the look-and-feel and for drawing the text. This is what the examiner believes to be a major claimed inventive feature of the claimed invention.*
- *The Appeal Brief filed 2-11-04, on page 5, from the third paragraph, "... the getDocument() and getStyledDocument() methods are not used for drawing the text attributable to a Text Component, but instead merely for accessing or fetching a document (or styled document) associated with the Text Component. These methods are embodied within the Text Component, and may be used by a Swing UI class to get the document (or styled document) when it is time for the Swing UI class to draw the Text Component. See, e.g., Nelson, page78, under the heading of "Interacting with Text Componets".*

Since the getDocument() and getStyledDocument() methods (as described on page 78 of Nelson) are not used for actually drawing the text, the methods cannot be considered equivalent to the presently claimed second software component." To this, the examiner agrees that it cannot be considered the second software component, because it is the first software component that defines the look-and-feel and does not draw the component.

The examiner will now address the individual arguments and statements made by the Appellant.

From page 4 of the Appeal Brief, from the first paragraph, the Appellant argues "Nelson, however, does not teach or suggest a first software component that is adapted to: (i) create a graphical representation of an object embodied by code, which includes text and other displayable content, and define visual attributes of the text without drawing the text."

The examiner contends that the Nelson reference teaches on page 78, a UI class (second software component) having separate groups of code to get the look-and-feel, and to draw the text, it teaches a UI class that doesn't know what the text control contains or what the contents should look like, but uses getDocument() or getStyledDocument() methods (first software component). As if further shown on page 73, the UI class (second software component) retrieves each Element from the Document (first software component) and creates a View that knows how to draw each

one. The Document is shown to contain visual attributes, on page 73, where Nelson teaches that the Document contains Elements that have Attributes defining how text should look. The appellant further admits in *the Appeal Brief filed 2-11-04, on page 8 and 9, from the last paragraph of page 8, "... the getDocument() and getStyledDocument() methods are not used for drawing the text attributable to a Text Component, but instead merely for accessing or fetching a document (or styled document) associated with the Text Component.*

From page 5 of the Appeal Brief, from the second paragraph, the Appellant argues "the Examiner appears to suggest that a Swing UI class may be used to "get the look-and-feel" of a text component, while the getDocument() and getStyledDocument() methods may be used for drawing the text attributable to the Text Component. Therefore, the Examiner appears to rely on a Swing UI class to provide teaching for the presently claimed first software component and the getDocument() and getStyledDocument() methods to provide teaching for the presently claimed second software component."

The examiner contends that the Appelant is incorrect in this assertion as can be see though the answer to the arguments in the final action where the Examiner pointed out that "Nelson teaches on page 78, UI class having separate groups of code to get the look-and-feel, and to draw the text. It teaches a UI class that doesn't know what the text control contains or what the contents should look like, but uses getDocument() or getStyledDocument() methods." The examiner, in this instance, interprets the UI class

to be the Second Software Component an the getDocument() or getStyledDocument() methods to be the First Software Component.

From page 6 of the Appeal Brief, from the second paragraph, the Appellant argues "the UI class of Nelson does not include 'code comprising text and other displayable content' ".

The examiner contends that the office interperets the UI class to be the Second Software Component an the getDocument() or getStyledDocument() methods to be the First Software Component. Where the appellant is correct, in this instance, that the UI class does not include 'code comprising text and other displayable content.' The getDocument() or getStyledDocument() methods, however, as shown supra do include 'code comprising text and other displayable content.'

From page 6 of the Appeal Brief, from the third paragraph, the Appellant argues "The appellants are confused as to how a 'constructor that is able to transfer a complete document from on text component to another' can be used to provide teaching for the presently claimed first software component. Transfering a complete document from one text component to another is not a claimed feature of the invention."

The examiner contends that the statement as presented is not relied upon for the teaching of a first software component but merely to show the transferring of information between components.

From page 7 of the Appeal Brief, from the third paragraph, the Appellant argues "There is no motivation to modify the teachings of Nelson to provide the presently claimed first software component".

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, the Nelson reference teaches on page 78 and on page 73, UI class (second software component) having separate groups of code to get the look-and-feel, and to draw the text, it teaches a UI class that doesn't know what the text control contains or what the contents should look like, but uses getDocument() or getStyledDocument() methods (first software component). The appellant further admits in t*he Appeal Brief filed 2-11-04, on page 8 and 9, from the last paragraph of page 8, "... the getDocument() and getStyledDocument() methods are not used for drawing the text attributable to a Text Component, but instead merely for accessing or fetching a document (or styled document) associated with the Text Component.*

From pages 7 (paragraph 5) through page 8 (paragraph 3) of the Appeal Brief, the Appellant argues "the combination of JtextField and Jlable components".

The examiner contends that this is not the combination of components that the examiner wished to treat as the First and Second Software components, in claim 1. The examiner, in this instance, interprets the UI class to be the Second Software Component an the getDocument() or getStyledDocument() methods to be the First Software Component.

From page 8 of the Appeal Brief, from the fourth paragraph, the Appellant argues "The Examiner has failed to adequately support and/or establish prima facie grounds of obviousness".

In response to applicant's argument that there is no teaching or motivation to suggest the aforementioned limitations of present claim 1, the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

Claim 17:

With respect to the group of claims including Claim 17, the Appellant's arguments are focused on the limitations regarding a peer component for redirecting a call between software components. More specifically, as stated from representative Claim 17, the limitation argued is:

> *a peer component adapted for redirecting a memory call to invoke text drawing methods of the second software component rather than text drawing methods of the first software component.*

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The Examiner asserts the limitation to be an element that redirects a call to invoke text drawing methods of the second software component rather than text drawing methods of the first software component

As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

> "*Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure*. In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997)."

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the references, are within the scope of these limitations.

- *Nelson teaches a system in which a UI class (second software component) draws text components, but doesn't know what the text control contains or what the content s should look link, so it gets the Document using the Text Component's getDocument() or getStyledDocument() method (first software component) (see page 78, under the "Interacting with Documents" heading). Nelson further teaches Documents containing Elements that comprise a mapping of Document characters to attributes, these attributes defining how a document text should look (see page 73 all). The "Interacting with the UI" heading further shows that "the UI class retrieves each Element from the Document and creates a View that knows how to draw each one."*

The examiner will now address the individual arguments and statements made by the Appellant.

From page 9 of the Appeal Brief, from the sixth paragraph, the Appellant argues "Nelson fails to provide teaching, suggestion, or motivation for a peer component, which redirects a call to invoke the text drawing methods of the second software component rather than the text drawing methods of the first software component".

The examiner contends that the Nelson teaches, on pages 694, 697, 472, and on page 72, a system for drawing text where one component can define attributes of an item and the actual displaying of the item can be implemented by another item. It is, however, a design choice, an item could just as easily be given attributes and drawn by the same software component. Similarly, the system of Nelson, on page 73, allows a software component to reference a Document to provide the Element and corresponding Attributes, but not to draw; and then directs the text attributes to the UI class, for drawing to the display. "Interacting with Document" on page 73, points out that a Document might be asked for the Element and returning. In this sense the Document acts as a peer component that access the Element an provides look-and-feel information back to the UI class.

From page 10 of the Appeal Brief, from the third paragraph, the Appellant argues "There is no motivation to modify the teachings of Nelson to provide the presently claimed peer component".

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, the Nelson

reference teaches on page 78 and on page 73, UI class (second software component) having separate groups of code to get the look-and-feel, and to draw the text, it teaches a UI class that doesn't know what the text control contains or what the contents should look like, but uses getDocument() or getStyledDocument() methods (first software component). The appellant further admits in *the Appeal Brief filed 2-11-04, on page 8 and 9, from the last paragraph of page 8, "... the getDocument() and getStyledDocument() methods are not used for drawing the text attributable to a Text Component, but instead merely for accessing or fetching a document (or styled document) associated with the Text Component.*

From page 10 of the Appeal Brief, from the fifth paragraph, the Appellant argues "The Examiner has failed to adequately support and/or establish prima facie grounds of obviousness".

In response to applicant's argument that there is no teaching or motivation to suggest the aforementioned limitations of present claim 17, the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

Claims 2 and 11:

With respect to the group of claims including Claims 2 and 11, the Appellant's arguments are focused on the limitations regarding a first software component for creating a graphical representation of an object, and define visual attributes, and a second software component that draws the text (as previously argued). More specifically, as stated from representative Claim 1, the limitation argued is:

> *a first software component adapted to create a graphical representation of an object embodied as code within the software component, wherein the code comprises text and other displayable content;*
>
> *an application program running under an operating system; and*
>
> *a second software component adapted for drawing the text, wherein the first software component is invoked during runtime by the application program to define visual attributes of the text, but not to draw the text, and wherein the second software component is invoked to draw the text using the visual attributes.*

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the references, are within the scope of these limitations.

- *Nelson teaches a system in which a UI class (second software component) draws text components, but doesn't know what the text control contains or what the content s should look like, so it gets the Document using the Text Component's getDocument() or getStyledDocument() method (first software component) (see page*

78, under the "Interacting with Documents" heading). Nelson further teaches Documents containing Elements that comprise a mapping of Document characters to attributes, these attributes defining how a document text should look (see page 73 all). The "Interacting with the UI" heading further shows that "the UI class retrieves each Element from the Document and creates a View that knows how to draw each one."

- *Guha teaches a system of rapidly displaying text as did Nelson but further teaches, in column 2, lines 14-45, placing characters in a bitmap in memory, where there pixels are combined into lines to reduce the amount of memory space required to store*

The examiner will now address the individual arguments and statements made by the Appellant.

From page 11 of the Appeal Brief, from the fifth paragraph, the Appellant argues that although Guha is not cited against the present claims 1 and 8, Guha cannot be combined with Nelson to teach or suggest the presently claimed first software component; and a combination would not overcome the deficiencies therein.

The examiner contends that there are no deficiencies to overcome and that the Nelson reference teaches on page 78, UI class (second software component) having

separate groups of code to get the look-and-feel, and to draw the text, it teaches a UI class that doesn't know what the text control contains or what the contents should look like, but uses getDocument() or getStyledDocument() methods (first software component). As if further shown on page 73, the UI class (second software component) retrieves each Element from the Document (first software component) and creates a View that knows how to draw each one. The Document is shown to contain visual attributes, on page 73, where Nelson teaches that the Document contains Elements that have Attributes defining how text should look. The appellant further admits in *the Appeal Brief filed 2-11-04, on page 8 and 9, from the last paragraph of page 8, "... the getDocument() and getStyledDocument() methods are not used for drawing the text attributable to a Text Component, but instead merely for accessing or fetching a document (or styled document) associated with the Text Component.*

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Dennis G. Bonshock
March 23, 2007

Kristine Kincaid
Supervisory Patent Examiner
March 23, 2007

Weilun Lo
Supervisory Patent Examiner
March 23, 2007

CONLEY ROSE, P.C.
P.O. BOX 684908
AUSTIN, TX 78768